SYSTEM AND METHOD FOR GENERATING GAMES AND TESTS

Summary of Certain Inventive Aspects

One aspect of the invention includes a game authoring system, comprising a first interface for generating a character, and a second interface for generating at least one place visited by the character.

Another aspect of the invention includes a game authoring system, comprising a game authoring tool for generating an electronic game, the electronic game including at least one character and at least one place that is visited by the character, and a game player for playing the electronic game.

Yet another aspect of the invention includes a game authoring tool for generating an electronic game, the game authoring tool comprising at least one interface for defining a player character, a place that is visited by the player character, at least one scene that is played when the user visits the place, and an interactive character having an associated dialog, the player character being able to access the associated dialog by visiting the place and speaking with the interactive character.

Detailed Description of Certain Embodiments of the Invention

Figure 1 contains a high-level overview of the inventions and processes. A computer **100** comprises content-creation software which may be used to create an interactive book, game, or test **104**. The creation process is described in more depth below with respect to Figures 2 – 14. The file created in book, game, or test **104** could be stored onto a disk, CD-ROM, or other storage device, or might be saved to hard disk and uploaded to FunEducation.com's Internet site **108**, where it could be distributed to others. Users, across the world, could then download the authored work **112**, and play the game, read the book, or take the test (depending on the file). The file might be free or sold to the user. In the case that it is a priced work, if the user tried to make a copy of it and send it to a friend, the friend would receive an evaluation version **116**, and be instructed as to where the work could be purchased.

Figure 2 describes some of the relationships between some of the databases in the software. (The table below gives more detailed information about the databases and their fields.) In table **200**, we begin with the Places database, which contains many fields, but we highlight only two of them (People Found, and What to Call). The fields for People Found could contain either an individual person or a Find Function (a group of People with similar characteristics). All of these are indexed links to the databases of People **204** and Find Functions **212**. The database for People **204** contains a number of fields, such as Name, Picture, Sound, and Person Characteristics **208**. These characteristics are given by the database for Person Characteristics **208**, where each characteristic can be defined dynamically by the Author, and can include strings, numbers, lists, indexes to People, and indexes to Places. In the database of Find Functions **212**, a set of preconditions **228** is listed that determines which people will be found.

Returning to the database of Places, another important field defines what will be called when the User goes to this place. Will it be a Scene (**216**), a Conversation (**232**), or a Test (not included in figure). If it is a scene, then this place will have a set of links that index different scenes from the Scenes database **216**. The scenes database contains a set of fields for displaying the scene to the User. Exemplary scenes are Parsed Text (which is the text that the user will see, but may include text that pulls information from the databases), Changes **220** (which describes precisely which changes will occur in the databases when this scene is called), and Links **224**. The Links database **224** contains the indices to future scenes that may occur, the chance that those scenes may occur, and the preconditions **228** necessary for them to occur. The Preconditions database **228** uses information in the People, Person Characteristics, Places, and Game Characteristics databases.

The final databases described in this figure are for Conversations **232**. Three databases make up what is called a conversation: Topics, Statements, and Responses. The Responses database **236** is similar to the others. Some of its important fields are for the Topic, Parsed Text, Preconditions, Percentage Chance, and Change. Each response in the database is associated with one topic, contains parsed text that will be displayed to the User, may have a set of preconditions that must be true for the response to occur,

has a percentage chance that it will occur (given preconditions are true), and has a set of changes that will occur to the databases if the response occurs.

Database Table	Field	Description
Places	Name	String name of Place
	Comments	Array of strings about describing Place
	Known by Hero	Boolean; if true, User can go to place
	People Found	Array of structures, each containing: Whether Find Function or Person; Index of Find Function or Person; Percentage Chance
	Multimedia	Path for image, video, sound, etc. file
	Sound	Path for sound file
	Go to what?	Integer: 0=Conversation, 1=Scene, 2=Test
	Links	Array of structures, each containing: Index for scene to go to Index for precondition set to use Percentage chance
	Topics	Array of integers, indexing which topics to use if place goes to a conversation
	Test Group	If place goes to test, index of which test group to pull questions from
	Test Changes	If User passes test, index of set of changes to apply
	Score to Beat	Score that User must have to pass test
People	Name	String, name of person
	Image	Path for picture of person
	Sound	Path for sound associated with person
	Individual Person Characteristics	Array of strings that will be same size as the number of person characteristics; string values are interpreted based on settings for each person characteristic
Person Characteristics	Name	Name of Characteristic
	Type	Is this characteristic represented by text, a number, a list, an index to a person, place, etc.
	List	Array of strings (if type is set to list)
	Default Text	String (if type set to text)
	Default Value	Numeric (if type set to number, list, or index)
	Maximum	Numeric (if type set to number)
	Minimum	Numeric (if type set to number)
	Introspectable	Boolean; can Hero know about this characteristic when Contemplating?

Database Table	Field	Description
Find Functions	Name	String name of function
	Precondition	Index of set of preconditions necessary for person to be included in group
	Group to pull from	0=All People; 1 or more = Index of Find Function
Scenes	Title	String title of scene
	Parsed Text	Text that User will see, including special characters that indicate formatting or the pulling of data from databases
	Multimedia	Path where image, video, or sound file is located that is included in scene
	Sound	Path where sound file is located that is included in scene
	Other Pic	Integer to tell software to use picture of Hero, Encountered Person, Place, etc.
	Other Sound	Integer to tell software to use sound of Hero, Encountered Person, Place, etc.
	Change	Index to set of changes to apply to databases when scene occurs
	Scene Links	Array of Structures including: Option text (that Hero will see) Precondition for text to arise Links to use if option selected (each link contains: Scene Index to use Precondition Index Percentage Chance)
Changes	What to change?	Integer specifying type of change (person, place, item, game characteristic, game over, etc.)
	First Item	Depending on what is being changed, this will index a particular database; for example, if a person is to be changed, first item will be the index of that person
	Second Item	Index, depending on type
	Third Item	Index, depending on type
	Fourth Item	Index or value, depending on type
	New Value	Index or value to set the database's element to
Group of Changes	Set of Changes	Array of Indices in Changes Table; this array defines the entire set of changes that will occur (called by Scenes, Responses, Place Tests, etc.)
Group of Preconditions	Set of Preconditions	Array of Indices in Preconditions Table; this array defines the entire set of preconditions that must be true for the set to be true (called by all tables that index preconditions; they are indexing this

Database Table	Field	Description
		table, and it indexes the preconditions table)
Preconditions	Who to check	Integer indexing person to check or game characteristic (= -10)
	Characteristic to check	Index of the characteristic being checked
	Comparison	How do you want to compare this value? (equal, not equal, greater than, less than, etc.)
	Compare to what?	Compare to a fixed value or another characteristic?
	What value?	Used for fixed value, or as index for a second person if comparing with another person's characteristic
	What characteristic2?	The other characteristic used in the comparison
Topics	Topic	String
Statements	Topic	Index
	Statement Text	Parsed text
	Precondition	Index of Group of Precondition
	Percentage Chance	Integer Value
Responses	Topic	Index
	Response Text	Parsed Text
	Precondition	Index
	Percentage Chance	Integer
	Change	Index to groups of changes
	Scene to call	If =0, none; otherwise, index to scenes
Contemplate	Allow user to contemplate	Integer; 0 = On himself and others, 1 = on himself only, etc.
	Characteristics to use	Integer
	Contemplate on others	Array of Structures, each containing: Name of function Initial Text Closing Text Find Function to run (Index)
Game Intro	Introduction	Parsed Text
	Multimedia	Path to multimedia file
	Image	Path to image file
	Sound	Path to sound file

Database Table	Field	Description
Game Characteristics	Name	Name of Characteristic
	Type	Is this characteristic represented by text, a number, a list, an index to a person, place, etc.
	List	Array of strings (if type is set to list)
	Default Text	String (if type set to text)
	Default Value	Numeric (if type set to number, list, or index)
	Maximum	Numeric (if type set to number)
	Minimum	Numeric (if type set to number)
	Introspectable	Boolean; can Hero know about this characteristic when Contemplating?
Events	Name	String
	Time Period	Integer; after how many time periods will this event occur
	How often	Every Time or only once
	Find Function or Person	Boolean
	Index of FF or Person	Index; database depends on Boolean above
	Change	Index
	Apply to	Boolean; Apply to All or Random Sample pulled from Find Function
	How many max	Integer; can't apply to more than this many
	How many min	Integer; must apply to at least this many
	Chance to apply	Numeric; percent chance that any one person pulled from Find Function will have occurrence of event
	Report News	Boolean; yes or no
	Headline	String
	Parsed Text for each person	Parsed text, pulling from database which people are affected
	Precondition	Index, determines which people will have articles written about them
Goals	Completed	Boolean; yes or no
	Precondition	Index; what must occur for goal to be considered completed
	Scene to call	Index; scene called when goal is first completed
	Hint	Text hint displayed to user to help him/her achieve goal
Warnings	Time Periods	Integer; after what time period will Warning appear
	How often	Every time or just once
	Preconditions	Index; what must be true for Warning scene to appear

Database Table	Field	Description
	Scene to call	Index; what scene is called if Warning appears
General Info	Author	String
	Date created	String
	Date last modified	String
	Contributors	String
	Author Bio	String
	Summary Description	String
	Audience Intended	String
	Credits	String
	Allow others to Edit	Boolean
	Free or Charge	Boolean
	Price	Numeric
	Distributable	Boolean
	Game Style	Integer; test, game, interactive book, etc.
	Who is Hero	Integer; created new, one of the characters, chosen from all, etc
	Heroes possible	Array of Integers; Characters that could be selected as Hero
	Help User with Characteristics	Boolean; if true, will show User the introspectable characteristics of each possible Hero to help him/her select the Hero
Items	Name	String
	Description	String
	Scene to call	Index
	How is it held	Integer; by person, in a place, randomly, etc.
	Which person/place	Index
	Precondition	Index to groups of preconditions
	Chance to arise	Integer
Interactions	Name	String
	Time Periods	Integer; after how many time periods will interaction occur
	How often	Every time or just once
	Find Function 1	Index; to find list for first group of people
	Find Function 2	Index; to find list for second group
	Chance of interaction	Numeric; percentage chance that interaction will occur for each pair
	Change 1	Index to group of changes; applied to person 1 if interaction occurs
	Change 2	Index to group of changes; applied to person 2 if

Database Table	Field	Description
		interaction occurs
	Report News	Boolean
	Headline	String
	Parsed Text for each person	Parsed text
	Preconditions	Index; to decide which people will have articles written about them
Visit	Allow user to visit	Boolean
	Find Function	Index; used to find people that are visitable
	Button Text	String that the Hero will see; might say "Visit", "Phone Call", etc.
	Chance to encounter person	Integer;
	Links	Array of structures, each containing: Scene Index Precondition Percentage Chance
	Allow for Conversation	Boolean
	Precondition	Index; used to determine whether conversation will occur
	Topics	Array of indices; what can be discussed
Time Period	Time name	String; examples: "day", "week", "light-year"
	Actions	Integer; the number of user actions that equate to one time period
Soundtrack	Title	String
	Credits	String
	Sound File	Path to sound file
Ads	Title	String
	Animated GIF	Path to animated gif file
	URL link	String
	Hypertext	String
Test General	Test type	Integer; defines how questions are ordered when displayed to user
	Test finish	Integer; defines whether user can see score and review answers immediately, or if he/she saves results to file
	Password	String; used for encryption

Database Table	Field	Description
	Display	Integer; defined things that are displayed to user while taking test
	Passing Score is	Integer
	Test Groups	Array of strings
	Test Time	Numeric; if test is timed, how many minutes do the users have
	AutoPass	Boolean; defines if Test will stop once student has achieved a passing score
Test Questions	Group	Index to test group
	Multimedia	Path to multimedia file
	Question	String
	Correct Answer	String
	Wrong Answers	Array of strings
	Explanation	String
	Difficulty	Numeric
	Value	Numeric
	If answered wrong	Integer; defines how to proceed with test if question is answered wrong
	Subtraction style	Integer; defines how many points will be subtracted from user's score if they answer the question incorrectly
	Student answer	Integer; used by reader software to store what the student has answered right, wrong, skipped, etc.

Figure 3 describes the steps of how to create a content file (test, game, or interactive book) and distribute it to be used by others. In **300**, the content creation software is opened, and in **304**, a file is opened or created new by the Author. (Some screen views are provided in Figures 4 and 5 showing the toolbars and menubars that an Author might see after opening a file.) In **308**, the databases can be edited in any order by the Author. (Figure 2 and the large set of tables that follow it give details of the workings of these databases.) In state **312**, the Author can play the file created with the Reader software. This might be done to troubleshoot, debug, or simply enjoy the work created. In state **316**, the Author finishes creating and testing the work, and converts it into a distributable file. A distributable file (or set of files) can be read and used **(320)** on other computers, and stored on various storage devices (CD-ROM, hard disk, floppy disk, internet site, etc.).

Figure 4 shows a screen display of the Authoring software after the Author has chosen to create a new interactive book or game. The menu bars give the Author access

to all of the databases, while the toolbar gives access to the most often used functions (including in this case, the People, Places, and Scenes Editors).

Figure 5 shows a screen display of the Authoring software after the Author has chosen to create a new test. The menu bars give the Author access to all of the databases used for testing, while the toolbar gives access to the most often used functions (including buttons for Test Properties, Groups, and Questions).

Figure 6 gives more detail for state **308**, showing a simplified grouping of the databases **600**. The Author can edit any database in any order. If the Author wanted to edit General characteristics of the file **604** (such as Author information, a summary of the file, the audience intended for, credits, a soundtrack, etc. **608**), he would use these database editors. The editors that can be used to create an interactive book or game (**612**) include People, Places, Items, Scenes, Time, Game Introduction, Hero, Events, Interactions, Responses, Topics, Statements, Game Characteristics, Person Characteristics, Goals, Warnings, Contemplation, Visit, Find Functions, etc. **616**. Finally, to create a test **620**, the Author would use the editors for Test Properties, Test Groups, and Test Questions **624**.

Figure 7 gives an example of the Authoring process. In **700**, the Author might enter in his name, a summary of the work, and his biography. In **704**, he chooses a group of songs (that are encoded digitally on his computer) that will make up the soundtrack of his work. In **708**, he types in the introduction that will appear to the user when the file is first opened, as well as including a sound effect, image, and video file in the introduction. In **712**, he begins typing in the characters of the game, specifying a name and picture file for each. He also uses the Person Characteristics Editor **(716)** to add in any other variables that he wants to keep up with for his characters.

Moving to the **720**, he begins to input the places of the interactive book, specifying a name, picture, and sound effect for each. Some of his places will be accessible to the hero at the beginning, and others will need to be discovered. He will specify all of that information here. He also specifies what each place will call. Some will call scenes, and the Author will write those scenes in **724** (including for each scene: the text that will appear, any additional multimedia, and any changes that will occur). Some places will call a conversation **728**, and the Author will need to use the Editors for

Topics, Statements, and Responses to define the possible conversations. Other places will call a test **732**, meaning that when the Hero goes to this place, he will have to take a test on specified material. The Author will specify how the test is set up in Test Properties **736**, the test group that will be used by this place **740**, and the test questions that will be asked in each group **744**.

Moving on to state **748**, the Author will set up the Hero of the game. Will the user pick from all characters or a subset of characters? Will the Hero always be the same character? Or will the user type in her name, causing a new character to be created?

In state **752**, the Author inputs special items that can be found by the Hero. Some may be held in specific places and only arise if certain preconditions are true. Others may be held by a random person, and arise 10% of the time when that person is encountered. The Author will be able to specify precisely when and under what conditions the items can be found.

State **756** lists some of the other databases that the Author might edit in the creation process. He might set up interactions between characters, events that occur, contemplation functions for the hero, etc.

Figure 8 is a screen display of one embodiment of the Places Editor. In this case, there are two places, Austin and San Diego, and the Author is looking at the details for San Diego. We see its name, that it is accessible initially, that no multimedia files are selected for it, that it calls one scene "San Diego Scene 1" (with no preconditions set, and a 100% chance). There could be many more scene possibilities for this place, but in this case, one scene will always be called when the Hero clicks to go to San Diego. Finally, there are two people that are set to be found in the place. (It should be noted that the buttons displayed change depending on what database the place calls. Since it is calling a scene, we see these buttons, but if it called a conversation or test, the buttons would change.)

Figure 9 is a screen display of the Person Editor. Here, the Author has three people set up, and is currently looking at the details for the character Susie. The Author can change her name, any multimedia files associated with her, and the characteristics currently set up for her (in this case, Age = 28, race = white, sex = female, and money =

15). Clicking on the button "Edit this Person" would allow the Author to change Susie's characteristics. Clicking on the button "Characteristics" on the bottom of the screen would bring up Figure 10.

Figure 10 is a screen display of the Person Characteristics Editor. Here, the Author has defined four characteristics: Age, a number; Race, a list; Sex, a list; and Money, a number. Since Age is highlighted, we can see the details for it. The Author can rename it (from "Age" to something else), can change it from a number to another data type, can change its default values, max and min, and can change whether it is introspectable. If we looked at Sex, we would see a list of two strings "male" and "female". Race might contain a list of five strings. In all cases, the Author has full power over decided which characteristics will be used, and how they will be represented. If they are lists, he can define which strings make up the list. Like all of the databases, there are no limits to the number of entries. The Author could make 250 characteristics for each person if desired.

Figure 11 shows a screen display of the Scenes Editor. In this case, the Author has created three scenes, and the current highlighted one is San Diego Scene 1. Looking at the details of it, we see an indicator displaying all databases that index this scene (in this case, it's only indexed by the place "San Diego"). The Author can modify the Scene Title, click on the button "Scene Text" and modify the text displayed to the user, click on buttons like "Multimedia" or "Changes" to define which multimedia files are used in the scene or which changes will occur when the scene is called. In this case, the Author has set up one link (using the "Links" button). This link defines the following: first, the User will see a text option for "See the zoo" when the scene appears; second, if the User clicks on "See the zoo", she will go to the scene "zoo" 100% of the time. Finally, there is no password set up for this scene. If one were set up, the User would be prompted with the text in the Password Prompt, and would have to answer the prompt with the exact word or phrase specified by Password.

Figure 12 shows a screen display for Parsed Text. Many of the databases use parsed text, which can include formatting information as well as links to the databases. In this case, the first line of the text says "You run into ~~Pe~Encountered Person~Name~~, who is walking in a very strange manner..." When this text appears

to the user, the part "~~Pe~Encountered Person~Name~~" will be replaced by the name of the encountered person. The software will pull the name from the databases and replace it in the text. From the Author's perspective, he simply needs to click on the button "Person", and specify "Encountered Person" and "Name" from two lists for "Which Person?" and "Which Characteristic?". It is important to note that the Author can choose this information from a menu instead of having to write it in like a programming language. It should also be mentioned that the parsed text can pull from all databases, not merely names from People.

Figure 13 shows a screen display for the Preconditions Editor. Here the Author creates a set of preconditions that are used in many of the databases. He can insert person preconditions (which use people and person characteristics) or game preconditions (which use game characteristics). In this case, he has specified that for this set of preconditions to be true, the Encountered Person's age must be less than 25, and the Hero's sex must be male.

Figure 14 shows a screen display where the Author is selecting a precondition that will go in the set. Using menus and lists, he can choose which person (the list includes Hero, Encountered Person, and all created characters), which characteristic (the list pulls from all person characteristics), which comparison (less than, equal to, etc.), how to compare (to a fixed value or compare with a characteristic of same or another person), and what to compare it with (the value or the person and characteristic). In this case, the Author has selected "Encountered Person" "age" "is less than" fixed value "25". Note that virtually every part of this GUI is dynamically filled with information from the other databases.

Figures 2-14 focused on the Authoring software and the process of making a content file (test, game, interactive book, etc.) Figures 15-38 will focus on the reader software, and how this file (which is basically a set of databases) is read and interpreted.

Figure 15 shows the high-level view of the reader software. The user will first open the file **1500**. The software will determine whether the file is a game or interactive book or a test **1504.** If it is a test only, the Test Reader software will be run **1508.** (Figures 16-20 show the Test Reader software in more detail). If it is a game or

interactive book, the Game Reader software will be run **1512**. (Figures 21-38 give more details on the Game Reader software).

Figure 16 shows the high-level view of the Test Reader. State **1600** displays the test groups and other information to the User. The User can then select a test group (which might also include a group of skipped questions) to test on **1604**. The software will then run a portion of the test for that group of questions **1608**. Afterwards, a decision state occurs **1612**, in which the user can continue the test and choose from other groups, or choose to exit. Depending on the test properties, the User will either see her score and be able to review answers, or she will be allowed to save the test to file **1616**. If saved to file, a teacher (possibly the Author) would be able to open the file or a directory of files (for the same test) and see student results **1620**.

Figure 17 shows a screen display of the Test Reader. Here we can see that the student has entered in his name, professor, and class, and has begun the test. In the lower right-hand corner, we see that there are nine total questions, one has been answered, four skipped, and four remaining. A perfect score is shown to be 100, and a good (or passing) score is shown to be 90. More importantly, there is a list box where the student can select which test group he wants to take. Here, we can see that he has already done the questions in the first group, and has selected the second group. Hitting the "Go" button will take him to a screen where the questions are asked.

Figure 18 describes the procedure for ordering and displaying the questions to the User. In **1800**, all test questions for the current group are filtered out. (The current group is the one selected by the User). In state **1804**, the questions are arranged in an order given by Test Properties (described in more detail in Figure 19). In state **1808**, the software asks a question to the user. State **1812** determines if the question was answered correctly, incorrectly, skipped, etc., and which question will be displayed next (more details in Figure 20). In decision state **1816**, the software determines if there are any questions left to ask. If so, it returns to state **1808** to continue the loop.

Figure 19 describes how the questions are arranged, using the setting in the database for Test Properties. In decision state **1900**, the database for Test Properties is used to determine the method for ordering the questions (this method was defined by the Author). If the method is "Order typed", then state **1904** will be called. In this case, we

don't change the order of the questions, and we set the opening question to be asked to be the first question in the group **1908**. If the method is "Intelligent Ordering", then state **1912** is called. Here, the software will sort the questions in order of their difficulty level (using the database). It will then set the opening question to be asked to be the middle question in the group **1916**. If the method is "Shuffle Questions", then state **1920** will be called, randomly shuffling the order of the questions in the group. State **1924** will then set the opening question to be the first question of the group.

Figure 20 shows the procedure for determining the next question. In state **2000** the software will determine whether the user answered correctly, incorrectly, skipped the question, etc. and will add to or subtract from her score. In decision state **2004**, the software will check the database of Test Properties to see if AutoPass is enabled. If so, and the User's score is above a passing score, the software will set a Boolean to stop the test **2008** and return. If AutoPass is not enabled, or the User does not have a passing score, then the software will continue to state **2012**. This decision state will determine what kind of ordering method is set up. If it is set for "Intelligent Ordering", then state **2020** will be called. If not, state **2016** will be called. State **2016** increments the test question to be the next one in the group. State **2020** will go to a more difficult question if the User answered correctly (meaning that it will move up in the group to find a more difficult question that hasn't been answered yet). If the user skipped or answered incorrectly, state **2020** will move to an easier question (moving down in the group to find an unanswered question). In both cases, if an unanswered question is not found moving in the set direction, then the software will find an unanswered question that is as close to the same difficulty as the previous. Moving on the state **2024**, the software will check to see if another question is left unanswered. In this state, the software will also check the test questions database to see if this question was answered wrong. If so, is the question set up to exit the group when answered wrong? If so, then state **2024** will return a "No" and move to state **2008**. Otherwise, the software will simply return whether there are questions remaining or not.

Figure 21 describes the high level flow of the Game Reader. State **2100** is the initialization state. In this state the User selects to either play a new game or open a previously saved game. If a new game is selected, then initialization will occur on the

characters (and their characteristics), places, items, goals, time period, etc. All controls and graphical interfaces will be set up for their initial state. (Note that most all of this information comes from settings made by the Author in the databases). After initialization, state **2104** is run. This is the main loop and will be described more fully in the next figure. When the User exits the main loop, state **2108** occurs, allowing the User to save his current state in the game.

Figure 22 describes the main loop of the Game Reader. In **2200**, the loop begins, and moves into **2204** to run the Time Algorithm (Figure 32). The Time Algorithm updates the time period, and checks for goals, warnings, events, and interactions. After the time algorithm, decision state **2208** is where the User selects an action. She might go to a place, contemplate, visit, or exit (among other things). If she selects to contemplate, then state **2212** will be run (Figure 26), giving her information about her character or other characters in the game. After contemplation, state **2200** will be called again. If she selects to visit, then state **2216** will be run (Figure 25), taking her to encounter the person chosen, either in a scene or conversation. After visiting, state **2200** will be called again. If she selects to go to a place, then state **2220** will be run (Figure 24). Here, she will go to a scene, conversation, or test defined by the place. Upon returning from the place, state **2200** will be called again. Finally, the User is able to exit at any time from this main loop, which would stop the game.

Figure 23 is a screen display demonstrating various aspects of the game. In this case, it is a game to teach students about Spanish and Madrid. Any ads would be shown in the upper-left-hand corner. (In this case, there are no ads, but if there were any, they would be pulled from the Ads database.) Directly underneath is a listing of places where the Hero can go. (These places were defined by the Author, and the only ones listed are those that are set to be accessible.) Underneath the places, there is a list of characters that the Hero can visit (in this case, the Author has defined the text to say "Phone Call"). (The people that make up this list are pulled from the Find Function defined for the Visit database.) Underneath the visit list is a selection box for contemplation. The User could select to contemplate on herself or using any of the contemplation functions that the Author might have defined. And underneath the

contemplation input is a text indicator showing a hint for the next goal. (This hint was pulled from the Goals database created by the Author).

On the right-hand side of the window, starting at the top is an indicator showing the current time period for this User's game. In this case, it is "Week 1". As the User goes to more places or talks to more people, time will increment (as defined by the Author). Underneath the time period indicator, there are buttons for the User to see the Introduction again, view the Credits, read the Newspaper, or stop the Soundtrack. Below these buttons, there is a picture of a bullfighter (which is the picture defined for the place "Plaza de Toros" – Spanish for "Bullfighting Stadium"). Because the User just highlighted this place (Plaza de Toros), this picture appears.

Figure 24 details the algorithm used when the User chooses to go to a place. First, in **2400**, the software will determine the person found in the place (more in Figure 36). In decision state **2404**, a check is done to see if a special item is found (more in Figure 35). If a special item is found, the scene is called for that item **2408**. (More on the scenes algorithm in Figure 27). If a special item is not found, then the decision state **2412** comes up. Here, the places database is used to decide what to do for this specific place. Does the software go to a scene, conversation, or test? After deciding, state **2416** is called for conversation (more in Figure 29), state **2420** is called for scene (more in Figure 27), and state **2424** is called for test in place (more in Figure 30).

Figure 25 details the algorithm used when the User chooses to go visit. First, in decision state **2500**, a check is done to see if a special item is found (more in Figure 35). If a special item is found, the scene is called for that item **2504**. (More on the scenes algorithm in Figure 27). If a special item is not found, then state **2508** is run. Here the Visit database is used to determine probability that the person will be encountered. It also ensures that there is a scene or conversation set up to go to. After calculating the probability, if the person is not encountered, then exit the algorithm. If the person is encountered, then move to decision state **2512**. Here, the Visit database is used to decide what to do for this specific place. Does the software go to a scene or conversation? After deciding, state **2516** is called for scene (more in Figure 27), and state **2520** is called for conversation (more in Figure 29).

Figure 26 describes the algorithm for contemplation. In decision state **2600**, the software determines whether the User selected to contemplate on herself or use a contemplation function. If she chose to contemplate on herself, then state **2604** is called, which will pull from the Contemplation, People, and Person Characteristics databases to determine how many characteristics to show and which ones are introspectable. The algorithm will then randomly display the number defined. If decision state **2600** finds that the User wanted to contemplate on others, state **2608** will be called. This pulls from the contemplation database the Find Function needed to run for this contemplation function. In state **2612**, we will then run the Find Function to get the group of people. In state **2616**, the software will display to the User the text defined by the contemplation function, including the list of people found.

Figure 27 describes the algorithm for going to a scene (as called in various places like the Go to Place, Go to Visit, and Time Algorithm). In the first state **2700**, the software will determine which scene to use. Some functions send a single scene index and others send a group of links. This state (described more fully in Figure 31) will determine one scene to use. Moving on to state **2704**, the Scenes database will be accessed, and the parsed text will be shown to the user, including any multimedia and options. Decision state **2708** asks if there are any scene options. If not, it will exit the algorithm. If there are options, it will wait for the User to select one of them. After selection, it will move to state **2712**, determining with option was selected and accessing the scenes database to find the Links associated with that option **2716**. It will then pass the Links back to state **2700** to begin the loop again.

Figure 28 shows a screen display for one of the scenes in the Spain game. This figure corresponds to state **2704** from Figure 27. In this case, the User has selected to go to "Tu casa" (Your home). We can see two images that are associated with this scene, as well as the scene text that is displayed. At the bottom of the screen, there is a list box showing the options for the User to pick from. In this case, the User has selected "Have a meal". (If she then hits the continue button, it will correspond to state **2712** in Figure 27.)

Figure 29 shows the algorithm for a conversation (which might be called by the "Go to Place" or "Go Visit" algorithms). State **2900** determines which statements the

User will be able to say. (This is described in more detail in Figure 37). State **2904** will then open a window to the User, describing the person encountered and listing the statement options. In state **2908**, the User selects one of the statements to say to the encountered person. State **2912** will then determine the response of the encountered person (more details in Figure 38). State **2916** displays the response to the User in a graphical window. Decision state **2920** checks the Responses database to determine if a scene needs to be called. If so, the appropriate scene will be called in state **2924** (more details in Figure 27). If not, the algorithm will exit.

Figure 30 shows the procedure for running a test when a place links to a test. In state **3000**, the software pulls from the Places database to determine which Test Group to use, and what is a passing score. State **3004** will then use the Test Group defined to pull out the Test Questions. State **3008** will then order the test questions using the ordering method defined by test properties (Figure 19 details this). State **3012** will then run the test (exactly like states **1808, 1812**, and **1816** in Figure 18). At the end of the test, decision state **3016** will determine if the user passed or not using the passing score set up in the Places database for this place and the score obtained from the answers to these questions. If she did not pass, the algorithm will exit. If she did pass, then state **3020** will apply the changes defined in the Places database for this place.

Figure 31 shows the procedure for finding a scene, using Links. In state **3100**, each link is checked to see if its preconditions are true. Those links whose preconditions are not true are filtered out. State **3104** pulls from those links whose preconditions are true. The software will access the probability for each one (using the percentage chance variable that each contains) and will randomly choose one of the links (after weighing them with their percentage chances). The algorithm will then return the scene selected.

Figure 32 details the Time Algorithm. State **3200** will increment the actions and determine if a new time period has occurred (using the Time Period database). State **3204** will check for any goals achieved. This process involves looking at all goals in the Goals database that have not been completed. Any of these which have their preconditions met will cause decision state **3208** to return a "yes" and invoke state **3212** which calls the appropriate scene for the goal. After the goals are checked, state **3216**

will check for warnings. The software will only check for warnings if it is a new time period. (For example, the Author might set up time periods to be in "Days" and 4 actions equals one day. In this case, warnings aren't checked after every action, but instead after every day, which occurs after every four actions.) So, if it is a new time period, the Warnings database will be accessed. For each warning that is set to occur on the given time period, its preconditions will be checked. If these preconditions are true, then state **3220** will send a "yes" to state **3224**, causing the appropriate scene to be called for this Warning. After checking all warnings, state **3228** will be run, which will check for events. The software will only check for events if it is a new time period (just like Warnings). The decision block **3232** will determine if the time is right, looking at all Events in the Events database and finding any that are supposed to occur. For each that should occur, state **3236** will be run, which will run the event algorithm (more in Figure 33). After checking all Events, the software will move to state **3240** to check for Interactions. This process is exactly like the one for events, only running if it is a new time period. Decision block **3244** will determine if the time is right, looking at all interactions in the database. For each interaction whose time is right, state **3248** will occur, running the Interaction algorithm (more in Figure 34). When this is done, the algorithm will exit.

Figure 33 details the Event Algorithm. State **3300** will determine which people are affected, using the settings in the Event database. In particular, the software will determine if the event happens to an individual or a group (Find Function). If it is set up for a group, the database will specify whether all people in group are affected, or a random subset. If it is a random subset, the database will specify the probability, maximum and minimum numbers. State **3304** will make changes to those people affected, using the change set up in the Event database. State **3308** will check if this event causes news, and if so, will check the preconditions for the news to occur. If both are true, then state **3312** will occur, writing an article in the newspaper, using the parsed text and headline defined in the event. When all people in the group have been checked, the algorithm will exit.

Figure 34 details the Interactions algorithm. In state **3400**, the software accesses the database to determine which Find Functions to run. It then creates two lists, one for

person one, and the other for person two. State **3404** will randomize the order of the two lists and match up pairs. The matching up involves taking the first person from each list and making them a pair, then taking the second person from each list, then the third, etc. until one of the list runs out of people. If any pair is made up of two of the same person, the pair is discarded. Any unmatched people are also discarded as we move to state **3408**. Here, we take each pair and check for an interaction, using the probability set up in the Interactions database. Using this probability, the software will randomly determine which pairs have an interaction and which don't. If the pair has an interaction **3412**, the software will apply the changes set up in the database. Change 1 affects Person 1. Change 2 affects Person 2. Decision state **3416** determines if news occurs using settings and preconditions in the database. If so, state **3420** adds an article to the newspaper. State **3424** ensures that all pairs are checked for an interaction. When all have been checked, the algorithm exits.

Figure 35 details how the software checks to see if a special item is found. First, it determines which place the Hero is in, and who the encountered person is. State **3500** scans the Items database to determine if any items are held by the encountered person or in the current place. If so, the group of possible items is passed to state **3504**, where all items whose preconditions are not true get filtered out. State **3508** takes those items remaining and uses the percentage chance variable defined in each item to determine which will be found. State **3512** ensures that if multiple items are left and found, only one will be picked. State **3516** then calls the appropriate scene for this found item, as defined in its database. The algorithm is then exited.

Figure 36 details how to determine the person found in a place. State **3600** uses the Places database to get the Person Found structure, which can include various people or Find Functions as well as a percentage chance for each. Decision state **3604** then uses all the percentage chances set up and randomly picks one of them. If a Find Function is selected, state **3608** is run, which will run the Find Function to get the list of people and randomly pick one person from the list. This person is then sent to state **3612** and the algorithm is exited. If instead, an individual is selected, then this person is sent to state **3612** and the algorithm is exited.

Figure 37 details how a set of statements is determined for use in a conversation. This algorithm might be called by the Go to Place or Go Visit algorithms. State **3700** will use the appropriate database (either Places or Visit) to determine which topics can be discussed in this conversation. State **3704** then begins a loop for each topic. First, state **3708** will pull out all statements from the Statement database that have the given topic. Next, state **3712** will filter and keep only those statements whose preconditions are true. State **3716** will then select one statement from those remaining, using the variable percentage chance to weigh its selection. Decision state **3720** then asks if any topics are left. If so, we return to state **3704**. If not, the algorithm is exited with a list of statements to use.

Figure 38 details how a response is determined. This is called in the course of a conversation, and the User has just selected a statement to say to the encountered person. State **3800** determines what topic that statement belongs to. State **3804** accesses the Responses database that pulls out all responses for the given topic. State **3808** will filter and keep only responses whose preconditions are true. Using that group, state **3812** will select one response, randomly picking one from the group after weighing them, using the variable percentage chance. In state **3816** the algorithm exits, returning with the response selected.

Figure 39 details the simulation database and properties for objects in a scene or conversation. State **3900** sets forth objects, including text, pictures, and buttons, that the User manipulates. State **3904** determines if an object should appear in a scene or conversation by checking to establish the truth of preconditions **3908**. State **3912** details the array of values that determine what to do if the object in state **3900** is selected. For example, if the User clicks on a picture in the scene, what should happen? Figure 39 defines a set of possible actions. Each item in the set is comprised of: A. a scene or conversation to go to or action to take, such as opening an application or moving to a hyperlink; B. a set of preconditions that must be true for this action to be possible; and C. a weighted number which is used to determine what percentage chance there is to take this action over other possible actions.

Figure 40 details the precondition database and defines what conditions must be true for a Precondition Set to be true. State **4000** shows the identifying set that can

contain any number of preconditions. State **4004** determines if all conditions in a set are true by comparing the variables in state **4008**. The variables in state **4008** can come from any number of databases, and can be compared to fixed values or other variables. For example, in Figure 14, the User is creating a precondition (through menu selections) that specifies that the encountered person's age must be less than 25. In Figure 13, the set of preconditions includes two preconditions: the one above, and the condition that the Hero must be male.

In a preferred embodiment, the User selects an object **4100** and simulation entries are retrieved **4104** as shown in Figure 41. For each possible link, the preconditions for that link are analyzed **4108** and the links that meet the preconditions remain as an option. For each remaining link, the chance number is evaluated and all chance numbers are summed in state **4116**. Then, each link is put in a range from X to Y, based on the chance number of the link, where X and Y fall between 1 and the sum, and no overlap between ranges as described in state **4120**. A random number is selected between 1 and the sum of chance numbers **4124** and the link with the range that contains the random number is selected in state **4128**. Afterwards, the link selected is used either by going to a scene or performing an action **4132**.

While the above detailed description has shown and described the fundamental novel features of the invention as applied in various embodiments, the disclosed embodiments of the invention are merely illustrative and do not serve to limit the scope of the invention set forth in the following claims. Those skilled in the art may practice the principles of the present invention without departing from the intent of the invention.

APPENDIX A

User Manual

Table of Contents

Chapter 1: Actions & Time

Perhaps you've noticed the timekeeper in the top, right-hand corner of the main window. It says "Action #", and each time you click to go read a chapter, it increments, from 1 to 2 to 3, etc.

As you play the tutorial game, you'll notice that it doesn't increment for everything. For example, if you click on "Game Intro" or "Game Info", the number of Actions stays the same.

Actions are the moves that a user makes in your game. When you are finished with this game, you will know exactly how many times you hit one of the "Go" buttons. (There can be as many as three 'Go' buttons).

In this game, we keep time with "Actions", but you may make a game that keeps time with "Day", "Week", "Light Year", "Nanosecond", or whatever your heart desires. You may also decide that the user needs to click "Go" various times before a new "Day" or "Week" occurs.

For example, maybe you want to keep time in months, and 10 user actions (10 times clicking 'Go') equals one month. In that case, you could go to ten places before "Month 2" appeared on the window...

To change the Time and Actions in your interactive book or game, use the menu 'Book/Game -> Necessary Editors -> Time'.

Chapter 2: Introduction

The "Introduction" is what you see when the Interactive Book or Game first runs. (For this tutorial game, it has an outline of the topics covered). The user also sees the Introduction when she clicks on the "Introduction" button of the Fun Studio Reader.

You may want your "Introduction" to wow the user with a cool video clip, or an interesting image, or a sound clip (or maybe all three). The text will tell them what the game is about, and what goals they have to achieve...

To set up the "Introduction", use the menu 'Book/Game -> Necessary Editors -> Game Intro'. Here you can add the appropriate text, as well as any multimedia.

Chapter 3: Credits

In the main window, you will notice a button labeled "Credits". Clicking on it, you will get lots of information about the game/book. This information includes:

Author
Summary
Audience Intended For
Date Created & Last Modified
Author Biography
Credits

This information is pulled from three editors in Fun Studio. In the menu, going to 'General -> General Info' allows you to put in the Author's name, summary, audience, etc. In here, you can decide whether the game/book will be priced or free, and whether others will be allowed to edit it. Allowing others to edit your game might be more meaningful for a teacher or professor, who is sharing their work with other colleagues. [NOTE: For these selections to take effect, you must make your game/book into a distributable format (*.fun format). Any file that is in a Fun Studio format (*.fs format) is always editable. To make a book/game/test distributable, you can use the menu 'File -> Make Distributable'.]

The second place where "Credits" comes from is the "Author Bio & Credits" editor, which is also located in the menu 'General'. Here, you can put information about yourself (you, the author), as well as any credits that must be given (use of multimedia, etc.).

The third editor is the Soundtrack editor (also located in menu 'General'). If you are including a soundtrack with your game/book, you should put the Title and Credits for each song. Fun Studio will then compile these credits with the credits that you put in "Author Bio & Credits".

Chapter 4: Soundtrack

A soundtrack is exactly what it sounds like. Your game/book can include any number of MP3-style digital songs that play in the background as the user navigates through your game/book.

From the menu, select 'General -> Soundtrack' to add the songs that you need. You should include a Title and Credits for each song, as well as browsing for the MP3.

Some important notes:

1. MP3 songs are compressed, but they are still big, averaging four megabytes per song. If you want to distribute your book/game, you should be cognizant of the filesize limitations of having a large soundtrack.

2. Songs are copyrighted, and there are legal issues with using copyrighted material in anything that you are distributing. Visit www.funeducation.com for any and all questions about legal issues with copyrighted works. Our web page is constantly evolving to help meet your needs. On the web site, there are many links to royalty-free multimedia, as well as multimedia that has associated royalties. The web site provides information on how you can include royaltied media in your distributed applications.

3. MP3 format is far better than WAV format. (WAV files of the same song tend to be 10 times bigger than an MP3 file of the song). You should never use WAV files in your soundtrack (due to their huge file size).

Chapter 5: People

People... The characters of your world. In Fun Studio, you can create all the characters that you want, and let them embark on a life of their own... There are two main ways to talk to them after you create them:

1. Going to a place that links to a Conversation
2. Going to a place that links to a Scene

In both cases, you must go to a place (like the chapter you are currently reading), and the place must reference the person found.

There will always be one, and only one, character found in the place. If the place links to a conversation (like the next place), then you will definitely encounter that person. If the place links to a Scene (like this place), then you will only encounter that person if the Author writes them into the scene text...

The key thing to remember is that you can write all the characters that you want into the Fun Studio, but until you link them to a place, and reference them in that place, they will not be seen by the user.

The next two places show examples of meeting people in Fun Studio. The first links to a conversation, and the second links to a scene. You should go to each of them various times, to see how they change on each trip. (Don't worry so much about the differences between scenes and conversations right now, as they are subjects that will be covered later).

Chapter 6: Person Characteristics

If you have been visiting the characters of the game and asking them questions, then you should have a relatively good understanding of Person Characteristics. To this point in the game, you may have talked with three different people:

1. Julie
2. Ted
3. Patrick

And you may have asked them about:

1. Their age
2. Their sex
3. Who they are?

In this case, all three of their answers came from person characteristics.

1. "Age" was a number, stored for all three characters.
2. "Sex" was a "list" (male or female), stored for each character.
3. "Who are you?" is a text answer, and each character had their own answer.

In the Fun Studio, you can make up any characteristics that you want. For example, if we wanted to add more information to each person, we could add the characteristics of 'Money', 'Country of Origin', 'Personality', 'Job Description', and 'Knows Hero'.

1. 'Money' would be a number. The range would be from 0 to 10 million (meaning that no characters could have more than 10 million dollars or less than zero). The default value would be 100, (meaning that characters start with 100 dollars, unless we specify otherwise). NOTE: You must enter in a default, and a MAX and min value for the number.
2. 'Country of Origin' would be a list. A list is a group of words, like 'America', 'Mexico', 'Canada', 'India', 'France', etc.
3. 'Personality' would also be a list. We might have 'Wonderful', 'Great', 'Good', 'OK', and 'Rude' as the words in our list. We then have five items, and we order them in a logical order (in this case, from best to worst). The section on Preconditions will explain why it is good to have our lists in logical order.
4. 'Job Description' would be stored as text (not a list or a number). We could write as much as we wanted as a Job Description for each person.
5. 'Knows Hero' may sound like a strange characteristic... It would be a list of two words, "yes" or "no". Each person in the game would begin with a "no"; in other words, they haven't met the hero yet. As the game goes on, there would be situations and scenes where they do meet the hero. In these cases, we would change their characteristic to be "yes". Once they know the hero, that might change the way that they interact with her.

These are all examples of People Characteristics. In your game or book, you can decide on the ones that you want and add an unlimited number of them.

Using Characteristics in a Scene

Now that you understand what person characteristics are, you may be wondering how you can use them in your game or book.

For example, when you ask Julie how old she is, she responds with "I am 30 years old." How does the Fun Reader know exactly what text to put in??

The secret is what is called "parsed text". As the author in Fun Studio, you write the text that will be read by the user... You write in the scenes, the conversations, everything... But imagine that you have 100 people, with 100 different ages. If the user asks one of them, "How old are you?" you don't want to have to write 100 different scenes.

So, in a case like this, you use "parsed text". The text that you put in might say:

~~Pe ~Encountered Person~Name~~ says to ~~Pe ~Hero~Name~~, "I'm ~~Pe ~Encountered Person~age~~ years old."

What the user reads is:

Ted says to Patrick, "I'm 45 years old."

The ~~ sign tells Fun Studio that it shouldn't output the text exactly. It should parse it. Fun Studio then reads the special text.

1. Pe - means a person
2. Encountered Person - is the person that was found in this scene.
3. The third word is the characteristic
   A. Name - means to output that person's name
   B. age - means to output that person's age
   C. the characteristic could be any that you have written.

NOTE: You don't have to type in any of the special text yourself. As the author, you have special buttons that add the text for you automatically. You simply pick which characteristic you want to add.

IMPORTANT: If you use special text with a person's name or characteristic, and you later delete that person or characteristic, you will need to find that special text and update it. Fun Studio does not update it automatically in the case of deletion.

Chapter 7: Game Characteristics

In chapters five and six, we covered Person Characteristics. Each person has a set of qualities that differentiate them from the other characters.

But what do we do in situations like the following?

1. We want to monitor the level that the hero has achieved. For example, in this game, we are monitoring your level by keeping up with the tests that you have passed. (You are currently at level 3.)

2. We might want to keep up with items that the hero has acquired, or certain things that have happened to the hero that might change the outcome of other scenes.

In either of these two cases, it wouldn't make sense to use a Person Characteristic, because we are only interested in one value (we don't need a value for each character). For these situations, we use Game Characteristics. These are just like Person Characteristics, except that we only store one copy.

Some examples:

1. In this tutorial game, we are using a Game Characteristic called "Level Achieved". It is a numeric, going from 1 to 5. Each time you pass a test, your level achieved goes up by 1. [Notice that we only need one copy of this characteristic. We don't need to keep up with Level Achieved for all the characters, because the hero is the only one that is important for this characteristic.]

2. Assume that we are making a game where the hero walks through a building. Each new area that she walks into will change the places that she can go to next. (For example, if she is on the 1st floor, then she could exit the building or get on the elevator, but if she is on the 14th floor, then she can't directly exit the building. She needs to first go to the 1st floor.) In this case, using a Game Characteristic called "Location" would be ideal. It would be a list (1st floor, 3rd floor, 14th floor, etc.), holding the current location of the hero.

Chapter 8: Preconditions

Preconditions are things that must be true for an event to occur.

For example, when I go to the bank and want to withdraw money, certain preconditions must be true.
1st: I must have money in an account in this particular bank.
2nd: The bank must be open (or they must have an ATM).

If both preconditions were true, then I could go into the bank (possibly wait in line) and make a withdrawal.
But what if they weren't both true...

1. If I didn't have an account with that bank, the teller might notify me nicely, and ask me if I wanted to open an account.
2. If the bank were closed, then I might walk up to the door, see that it is closed, and get back in my car.

Using Preconditions in Places

Let's use the same banking example from the last chapter.

If we were to implement this kind of logic into the Fun Studio, then we would set up the following...
A. Game Characteristics
1. Money in bank - Numeric - 0 to $1 million
2. Bank is open - List - (true or false)
3. Bank used - List - (Wells Fargo, Bank of America, etc.)
B. Places
1. Wells Fargo
C. Scenes
1. Wells Fargo is closed
2. Wells Fargo - you don't have an account
3. Wells Fargo - withdraw money

In this example, there is only one place where the hero can go (Wells Fargo), and three possible scenes that could be called.

We would use preconditions with this place. We would say:

1. When the hero goes to Wells Fargo, and "Bank is open" = False, then go to scene "Wells Fargo is closed".
2. When the hero goes to Wells Fargo, and "Bank is open" = True, but "Bank used" does not equal "Wells Fargo", then go to scene "Wells Fargo - you don't have an account"
3. When hero goes to Wells Fargo, and "Bank is open" = True and "Bank used" = "Wells Fargo", then go to scene "Wells Fargo - withdraw money".

Do you see how the Game Characteristics are used here?? They are used to construct sets of preconditions. ("Bank is open" = True; "Bank Used" = "Wells Fargo"). And when these preconditions are true, then the appropriate scenes are called.

Preconditions can contain both Person Characteristics and Game Characteristics. You can use various logical phrases, such as "equal to", "not equal to", "greater than", and "less than".
NOTE: The one thing to remember is that all of the preconditions in the set must be true for the set to be true!! (If you have a set of four preconditions, and three of them are true, then the set will be false. All four must be true for the set to be true.)

Chapter 9: Places

The places in Fun Studio may not seem intuitive at first.
A place in Fun Studio is an item that the hero can click on and go to. Some examples:

1. The bank
2. The airport
3. The large building
4. The fourth floor
5. Upstairs
6. Outside
7. The Water Cooler

Some not-so-intuitive place names:

8. Test on Chapters 1-4
9. Visit the Monk
10. Chapter Nine: Places
11. Throw the Javelin

So you see, a place in Fun Studio is not always a place in real life. It might be a verb phrase, like "Throw the Javelin". It might be a chapter heading or a test, like the places in this tutorial game.

Places share common qualities.

1. The place name is the phrase that the hero sees and chooses in the game.
2. All places call something else: A scene, test, or conversation.
3. Places can contain multimedia (an image of the place, a sound heard in the place).
4. Places can (and often do) contain people. (Refer to chapters five and six).
5. When a place calls a scene, it uses a Link object (talked about in the next chapter).

NOTE: Multimedia associated with a place will not always be seen or heard. When linking places to scenes, you can set up the multimedia of the scene to use multimedia from the place. (When linking to a conversation, the place's image will automatically be used. When linking to a test, only the image of the place will ever be used.)

Chapter 10: Links

Links in places tell Fun Studio what to do when the hero clicks to go to a particular place.

A place can link to either:
1. A scene or set of scenes
2. A conversation
3. A test

Links are used with scenes. They allow you to call different scenes for the same place (using preconditions and percentage chance) or just call one scene for a place.

Let's use the banking example from before.
1. The place is named "Wells Fargo"
2. There are three possible scenes, each called based on preconditions.

The links for the place "Wells Fargo" would look like the following:

1. Place calls a scene (as opposed to a conversation or test).
2. Links
   A. Go to: "Bank Closed" scene
      Precondition: "Bank is open" = False
      100% chance
   B. Go to: "You don't have an account" scene
      Precondition: "Bank is open" = True
                    "Bank Account" does not equal "Wells Fargo"
      100% chance
   C. Go to: "Withdraw Money" scene
      Precondition: "Bank is open" = True
                    "Bank Account" = "Wells Fargo"
      100% chance

NOTE: One new thing to notice is the percentage chance. This is the probability of going to the scene, given that the preconditions are met. In this case, there could never be more than two possible scenes, but let's explore the example below.

Assume the following scenario:
1. Place = "Board Room"
2. Scenes = 1. "Board Room - empty"
            2. "Board Room - meeting going on"
            3. "Board Room - people eating lunch"

When the hero clicks to go to Board Room, we could set up the following links.

Links
A. Go to "Board Room - empty"
   Preconditions: none
   50% chance
B. Go to "Board Room - meeting going on"
   Preconditions: none
   25% chance
C. Go to "Board Room - people eating lunch"
   Preconditions: none
   25% chance

Now, we have three scenes that are possible every time (since no preconditions are set for any of them). When the hero clicks to go to the "Board Room", half the time it will be empty. 25% of the time, there will be a meeting, and the other 25%, there will be people eating.

Now, consider what would happen if we changed the percentages... What if they didn't add up to 100% (they might be more or less).

What would happen in this case?

A. "Empty"
   30% chance
B. "Meeting"
   15% chance
C. "Eating"
   15% chance

In this case, the percentages don't add up to 100%. Therefore, the Fun Studio would automatically adjust them when the game was running. Once adjusted, they would be the same as the 50-25-25 in the first example.

How about this case?

A. Empty - 100%
B. Meeting - 100%
C. Eating - 100%

In this case, they would be adjusted again, and each would have the same likelihood (approx 33% each).

One more example...

Consider that we add:

A. Game Characteristic
   1. "Time of Day" = List = "Morning", "Noon", or "Night"

We then would adjust the links.

Links

A. "Empty"
   Precondition: none
   50% chance
B. "Meeting"
   Precondition: none
   25% chance
C. "Eating"
   Precondition: "Time of Day" = "Noon"
   50% chance

What happens now?

1. We have added a Precondition for the "Eating Lunch" scene. So, if "Time of Day" = "Noon", then all three scenes will be possible. If "Time of Day" does not equal "Noon", then only the first two scenes will be possible.

2. If all three scenes are possible, then the probabilities will equal 125% total. Therefore, Fun Studio will adjust them down. In this case, they will be adjusted to 40%-20%-40%.

3. If it is not noon, and only the first two scenes are possible, then the percentage chance only equals 75% total. So, again, Fun Studio will adjust the percentages, this time to be 66% chance "Empty", and 33% chance of "Meeting".

Chap 11: People Found

Remember all the people that you met and talked to in chapters five and six?? Now we'll discuss how that was achieved.

There are two main ways to talk to characters:

1. Going to a place that links to a Conversation
2. Going to a place that links to a Scene

There will always be one, and only one, character found in the place. If the place links to a conversation, then you will definitely encounter that person. If the place links to a Scene (like this place), then you will only encounter that person if the Author writes them into the scene text...

When you set up a place, there is a button for "People Found". If you don't select anything for people found, the place will randomly pick one character from all of the "People" in the game.

If you click on the button, you have a pop-up window, which lets you add in all of the people that you want, as well as the percentage chance that they would be in the place.

For example, in this game, the place "Talk to Characters in a scene" has the following people found:

1. Julie (80% of the time)
2. Ted (20% of the time)

So, 80% of the time that the hero goes to this place, Julie will be there. If she is not there, Ted will be.

You can define whoever you want to be found in a place. But remember, if the place calls a scene, and you do not reference those people in the scene (remember the special text?), then the hero won't find them.

Chapter 12: Find Functions

What happens if your game is really dynamic (changing). What if you want your characters to be moving around a lot, and they won't always be in the same place.

For example, what if we make a game about your workplace, and Julie and Ted are often found at the water cooler when the game starts... But what if Ted gets fired, and/or Julie gets a promotion and moves to another building. It no longer makes sense for Ted to be at the Water Cooler, since he's no longer with the company. And Julie shouldn't be there either, since she is now in a different building...

So what do we do??

In a case like this, we would use a Find Function. In Fun Studio, a Find Function will find a set of people that have certain characteristics (remember the preconditions?).

For example, we might have a Person Characteristic:

1. Workplace = List (None, Building 1, Building 2, etc.)

At the beginning of the game, both Ted and Julie work in Building 1. But as the game goes on, Ted gets fired (now he works in "None"). Julie gets promoted, and now works in "Building 2".

Do you see where we're going??

So we would have a Find Function called:

"Building 1 Employees"

Looks through All People, and finds those whose:

"Workplace" = "Building 1"

Then the place Water Cooler would have People Found:

1. Find Function, "Building 1 Employees", 100%

By using the Find Function, we can achieve our goal: Julie and Ted are found there while they work in Building 1, but as the game goes on, other employees will be found there.

Using Find Functions in Places

So, let's review.

1. Find Functions allow Fun Studio to find people who match certain preconditions.
2. Places can use Find Functions so that characters can move around dynamically in a game.

Some Notes:

A. Make sure that if you use a Find Function in a place, that there will always be at least one person found. Otherwise, the hero will encounter a nobody in the place (and you may have written the scene to expect somebody there).

B. Find Functions are a more advanced topic. They don't have to be used to make a game.

C. Find Functions can call other Find Functions. For example, I might have a function to find all the teenagers in the game:

1. Function 1: "Teenagers"
   Pull from: All People
   Precondition: Person's "Age" is greater than 12
   Person's "Age" is less than 20
2. Function 2: "Teenage Athletes"
   Pull from: "Teenagers"
   Precondition: Person's "Plays Sports" = True

Notice how the 2nd function called pulled people from the first. By doing this, we are saving programming steps and also making our game easier to read by another Author.

Chapter 13: Scenes

Scenes -- the meat of most games... Scenes include the following key components:

1. Scene Title
2. Scene Text
3. Multimedia used
4. Changes that will occur
5. Links
6. Passwords

1. The Scene Title is the name that you give the scene to help you remember it (as the author). The user will never see this. The scene titles are then put into a list that the other editors can access. (For example, when you use the Places editor and link a place to a scene, you will pick the Scene Title to link to).

2. The Scene Text is what you are reading right now. It is a combination of regular text and special text (discussed in chapter six, and the following chapter). This is the text that the user will see.

3. Multimedia used. Any images, videos, or sounds that you want the user to see or hear when this scene is called. You might include pictures of the hero, encountered person, place, etc.

4. Changes. These are changes to characteristics in the game that will occur when this scene is called. For example, if you want to make places accessible or not, change person characteristics or game characteristics, or change the location of an item. A change might also be to cause a Game Over.

5. Links for Scenes are similar to links for Places. The only difference is that Scene Links have Link Text associated with them. More in a following chapter.

6. Passwords are rarely used in Scenes, but are appropriate for certain situations. For example, if you wanted a scene to be accessible only if the user could answer a specific question. The password is the answer, and the Password Prompt is the question. This scene links to a password protected scene. If you get the password correct, you continue on.

Chapter 14: Scene Text

Remember the section of parsed text (just after chapter six)??

Scene Text is parsed text, a combination of regular text and special text. The special text links to the database of people, places, things, and game characteristics.

The following is an example of special text being used...

Your name is p. You have come to this place and encountered Ted... Ted is male, 45 years old.

If you don't remember the concepts above, please go back and review the place after chapter six "Using Characteristics in a scene".

Chap 15: Changes

Changes can occur in various places, but most often occur in Scenes. The following is an example of the changes that happened when you passed Test 2.

Changes:
1.) 'Chapter Seven: Game Characteristics' is accessible to hero.
2.) 'Chapter Eight: Preconditions' is accessible to hero.
3.) 'Using Preconditions in Places' is accessible to hero.
4.) 'Chapter Nine: Places' is accessible to hero.
5.) 'Chapter Ten: Links' is accessible to hero.
6.) 'Chap 11: People Found' is accessible to hero.
7.) 'Chap 12: Find Functions' is accessible to hero.
8.) 'Using Find Functions in Places' is accessible to hero.
9.) 'Test 3' is accessible to hero.
10.) The Game's "Level Achieved" = 3.00

In this case, these changes happened because you passed a test. The Place-Test object made these occur.

But the same can occur in any scene. For example, let's assume that you have two Game Characteristics:
1. Money - Numeric
2. Apples - Numeric

We have a place that calls "The Store Scene". (The next place will demonstrate this...)
When you go to this place, the first scene will ask you which you want to do:
1. Buy an apple (if you have money)
2. Sell an apple (if you have an apple)
3. Leave store

When you choose one of these, you are choosing Link Text, which will take you to a link (calling the next scene).
Three scenes are possible:
1. Buy apple scene
2. Sell apple scene
3. Leave store scene

If you buy an apple, it will call "Buy apple scene", which will make the following changes:
1.) Money = Current Value - 1
2.) Apples = Current Value + 1

This will subtract one dollar from the hero, and add one apple to his holdings.

If you sell an apple, it will call "Sell apple scene", which will make the following changes:
1.) Money = Current Value + 1
2.) Apples = Current Value - 1

If you choose to "Leave Store", then no changes will occur.

Try this out in the next place. Also note that parsed text is used to tell you how many you have left. Go there various times. Notice what happens if you have no dollars or no apples.

Scene Links

Scene Links... the network of your game. These links tell Fun Studio how to link scenes to other scenes...

Did you notice in the last place (Apple Store) what happened when you had no dollars or no apples?? If you had no dollars, you weren't offered the possibility of buying apples. And if you had no apples, you weren't offered the possibility of selling them...

How did we do that??

Each scene has its own set of links. A Scene Link includes the following:

1. Link Text (the options that you saw: ie. "Buy Apples")
2. Preconditions (for the option to appear)
3. Links (scenes to call if this option is chosen)

In the Apple Store example, there were three Scene Links.

A. Link Text = Buy Apples
   Precondition = "Money" is greater than 0
   Link = Go to "Buy Apples Scene"
      Precondition: None
      100% chance

B. Link Text = Sell Apples
   Precondition = "Apples" is greater than 0
   Link = Go to "Sell Apples Scene"
      Precondition: None
      100% chance

C. Link Text = Leave Store
   Precondition = None
   Link = Go to "Leave Store Scene"
      Precondition: None
      100% chance

Notice what is happening here. There are three Scene Links, A, B, and C. Each has its own set of Links (the same kind of Links that a Place has). The difference is that Scene Links also has:

1. Link Text
2. Precondition for it to appear

The Link Text is what the user will see. It is an option for him to select from. The precondition ensures that only appropriate Link Text appears. In this case, if you don't have any money, you can't buy apples. Therefore, you don't even have the option. (If the precondition isn't met, the text won't appear).

When Link Text is selected, that's when the Links object is used. This is just like Place Links. There can be multiple scenes, and each scene can have its own preconditions and percentage chance.

Chapter 16: Picking the Hero

Any Game or Interactive Book must have a hero. The user of the game will be that hero, and will interact with the places and other characters as the hero.

In Fun Studio, you have many options when deciding on the hero.

1. The Hero can be one of the characters
2. The user can pick the Hero from all characters
3. The user can pick the Hero from a select group of characters
4. The user can type in her name, and the Hero is made as a new character

All of these things are defined by you with the Hero Editor. (Book/Game -> Necessary Editors -> Hero). You will choose one of the four alternatives.

1. As the Author, you might select one (and only one) of the characters to be the hero. Perhaps this is a character that you have defined to have certain person characteristics that are important for the Hero to have at the beginning. And, in this case, the Hero will always have the same name.

2. You might want to allow all the characters to be possible Heroes. In this case, each time that a user starts a new game, she can select from any of the characters and pick the Hero.

3. What about the case where you have 30 characters, but only two or three are Hero potential? In this case, you would select those two or three in the Hero Editor as possible Heroes. When the user starts a new game, she can pick from one of them (and only them).

4. The final option is to let the user type in her name, and a new character will be created for her. This character will have person characteristics equal to the default values that you entered in the Person Characteristics Editor.

NOTES:

A. Obviously, the hero can never encounter herself in the game. For example, assume that one of your characters is Julie, and she is commonly found in "The Restaurant". Julie is also a possible Hero. When a user starts your game and picks Julie as the Hero, you don't have to worry about Julie being encountered in "The Restaurant".

B. It is common for there to be two or three possible heroes, each with differing characteristics. (One might be male and another female, for example).

C. If you want your hero to have randomized characteristics at the beginning, then don't use alternative 4 (Make new hero). This is because alternative 4 uses the default settings, which cannot be random.

Hero Example

Remember when you first ran this tutorial game, and it asked you to type in your name?? You were making a new Hero, and the name you typed in became the Hero's name. Her person characteristics came from the default settings of all person characteristics.

Below is information about yourself (the Hero). You can see this if you contemplate. These characteristics are "introspectable" (a setting in both Person and Game Characteristics Editors). An "introspectable" characteristic is one that you can think about when you contemplate. These characteristics might also show up to help you pick the Hero (used only when the user can select the hero from two or more possibilities).

Chapter 17: Goals

One way to help the user is by inserting a lot of sub-goals into your game. Since you can give a hint for each sub-goal, the user can better know what she should be doing. Thinking about the goals will also help you create the flow of your game.

Goals in Fun Studio are made up of the following pieces of information:

1. Scene to call once the Goal is achieved
2. Preconditions to determine if Goal is achieved
3. A hint to help the user achieve the Goal

1. One (and only one) scene will be called when a Goal is achieved. If you haven't already defined the scene (in the Scenes Editor), then the Goals Editor will let you name the Scene. After you name it and save the Goal, the new scene will be created. (You will need to go to the Scenes Editor at a later time in order to add the Scene Text, Links, and any changes that you want to occur).

NOTE: The scene will only be called once... It will be called when the user has achieved the goal, and is returning to the Main Screen.

2. Preconditions are used to determine when the Goal is achieved. Hopefully, you remember Preconditions from the first tutorial. But, just in case, we will review them now.

A. A precondition is a statement (using person or game characteristics)
B. This statement can either be true or false
(examples: Julie's "Age" is greater than 30.
The Hero's "Citizenship" = American.
The Game's "Level" does not equal 3.)
C. You use a set (one or more) of preconditions, and all must be true.
D. In the case of Goals, if all in the set are true, the goal is achieved.

3. The Hint. In the Goals Editor, as you select the preconditions and the scene to call, you should also type in a hint for the user. If you type one in, then it will show up on the screen as the Hero is playing, to help direct her through the game. Once the goal has been achieved, the next hint (of the next goal in the list that is not achieved) will be displayed.

NOTE: You should try to put your goals in a logical order, but sometimes a later sub-goal will be achieved before an earlier one. For example, if you have 5 sub-goals, and the Hero begins the game, the hint for the first one will be displayed on the screen. But it is possible that the Hero achieves the second one (or even higher) before achieving the first. In this case, the scene will be called for the second one. And later, when the Hero achieves the first, the next hint will come from the third goal (since the second was achieved).

Chapter 18: Warnings or Game Over conditions

Imagine the following situations. What would you do in each?

1. You want the game to end after 100 days.
2. If the Hero hasn't gotten to level 2 after 10 weeks, you want to tell him to do better.
3. If the Hero hasn't gotten to level 2 after 20 weeks, you want the game to end.
4. Every 5 days, if the Hero has no money, tell him to get a job.

The answer for all three is to use the 'Warnings/Game Over' Editor. This Editor will need four pieces of information.

A. When does the warning occur? (after 10 weeks, 100 days, etc.)
B. How often? (every 10 weeks; or just one warning after week 10, and none after week 20, 30, etc.)
C. What preconditions must be true for the warning to happen?
D. What scene to call if the warning happens?

So if you were trying to achieve #1 above (the game to end after 100 days), then you would set up the following:

A. After 100 days
B. One time only
C. No preconditions
D. Call "Game Over scene" (and this scene would have a change set to "Game Over")

NOTE: If you want a warning to cause a Game Over, you should set that up as a change in the scene. The 'Warnings/Game Over' Editor doesn't set it up. Only in the Scenes Editor can you set it up.

Next Example, #2 (If the Hero hasn't gotten to level 2 after 10 weeks, you want to tell him to do better). In this case, you would set up the Warning as the following:

A. After 10 weeks
B. One time only
C. Precondition = The Game's 'Level Achieved' is less than 2.
D. Call "Warn Hero to do better scene"

Example #3 (If the Hero hasn't gotten to level 2 after 20 weeks, you want the game to end). In this case, you would set up the following:

A. After 20 weeks
B. One time only
C. Precondition = The Game's 'Level Achieved' is less than 2.
D. Call "Game Over scene" (again: this scene would have a change set to "Game Over")

Example #4 (Every 5 days, if the Hero has no money, tell him to get a job). In this case, you would set up the following:

A. After 5 days
B. Every time
C. Precondition = The Hero's money is less than 1.
D. Call "Warn Hero to get a job" scene

Chapter 19: Testing in a 'Place'

You just finished taking Test 1 to be able to see Chapters 4 & 5. Let's talk about how this was achieved with Fun Studio...

You might remember that each place can go to one of three things:

A. Conversation
B. Scene
C. Test

You set this up in the Places Editor, and in this case, choose for the place to take you to a Test. You can then click on the "Test" button to set up the properties. You will need to put in the following information:

1. Which Test Group to use? (determines which questions are asked)
2. What is a passing score? (somewhere between a minimum score and a perfect score)
3. What changes to make if the Hero passes?

If you haven't defined Test Groups and Questions, there are buttons in the Editor to let you do so. Remember that each question can be in one (and only one) Test Group.

When determining a passing score, remember the following:

1. The Editor will show you what a perfect and good score is for the group.
2. Your score is for this Test Group only; not for all of the questions.
3. The Hero's score must be higher than the passing score! So if you set the passing score to be 69, the Hero must make a 70 or higher. If the Hero makes a 69, he will not pass, and the changes will not occur.

The changes that you set up here are the same that you would set up for a Scene. Some examples might be:

1. Make the next place accessible.
2. Make this current place (the test) inaccessible.
3. Make the 'Level Achieved' one point higher.

Chapter 20: Items

Items are special goodies that the Hero might find in the world. They can be located in a place (fixed or random), or held by a person (fixed or random). As the Author, you define under what conditions these items can appear, and what scene will be called when they do.

The Items Editor needs the following information:

1. Item Name
2. Item Description
3. How is it held? (by a Person, Place, randomly, etc)
4. Preconditions that must be true for it to appear?
5. Chance that it will appear, given that preconditions are true.
6. Scene to call if it appears.

Let's do a quick example. Let's say that the item is a "Five-Dollar Bill", and it appears in the "Living Room" (under the couch) 10% of the time. We would define this item in the following way:

1. Name = "Five-Dollar Bill"
2. Description = "A brand-new five-dollar bill, poking out the side of the couch."
3. Held in fixed place = "Living Room"
4. Preconditions = None
5. Chance = 10%
6. Scene to call = "Find five dollars" (in the scene, make a change that Hero's 'Money' = Old Value + 5)

Let's discuss an advanced example of an item. Let's say that there is a 'Special Coin' in the game that will be an item. This coin can be found in the Master Bedroom, but will only be found there 60% of the time, and only if the Hero is wearing the "secret glasses". We would set this up as:

1. Name = "Special Coin"
2. Description = "A shiny coin, which looks to be centuries old."
3. Held in fixed place = "Master Bedroom"
4. Preconditions = The Hero's 'Wearing secret glasses' = True
5. Chance = 60%
6. Scene to call = "Find special coin" (in the scene, make a change that this item is now held by Hero, and not in Master Bedroom)

NOTES:

A. The Item will never be found if the preconditions aren't true. And if they are true, then the Item will only be found x% of the time, defined by #5 (Chance).

B. If you want the item to not be found more than once, then you'll need to set up a change in the Scene. The change would be "Item is held by Hero". If you don't set this change, Fun Studio will think that the item is still in its old place, and you will be able to find it multiple times.

Chapter 21: Conversations

Conversations can be used instead of scenes, when you want the Hero to be able to ask questions to different characters in the game. It is easier to set up these question and answer sessions as conversations than as scenes. You'll see this in the following examples.

Example 1. Assume that you want to set up a place where 10 different people could be encountered, each of them quite distinct. When you go to the place, you want to be told a little about the place and the person encountered. You then want to be able to ask questions to the person, and you want those questions to be appropriate for that person. For example, if you encounter a good friend, you might say, "Wazzup?!?", but if you encounter the CEO of a major corporation, you would say, "Good morning, sir" instead. And if the CEO was a woman, you would say, "Good morning, ma'am". Etc...

It would be difficult (and in some cases, impossible) to set up intelligence like this using scene links and link text. That's why the Conversation objects exist.

A conversation is broken into three parts.

1. Topics - what the Hero can talk about with the Encountered Person
2. Statements/Questions - the phrases that the Hero can say
3. Responses - how the Encountered Person responds to the Hero

The next three sections cover each one of these in detail.

NOTE: When you go to a conversation from a place, the opening text will describe the place and the person encountered. The way that Fun Studio knows how to describe the place is by using the Comments that you define for the place. "Comments" will appear as an extra button in the Places Editor, and the comments that you type in are used to describe the place when the Hero goes there (to have the conversation).

A. Topics

Remember the example in the previous chapter? The Hero might say "Wazzup?!?" or "Good morning, sir" or "Good morning, ma'am". All three of these are 'Statements' of the same topic. The topic in this case might be "Greeting".

Topics are general concepts of conversation. Examples are:

1. "Greeting"
2. "How's the weather?"
3. "How are you?"
4. "What's your name?"
5. "Can you help me with ..."
6. Etc.

Each of these topics could have various statements or questions, depending on who the Hero was, and to whom she was speaking. We already discussed the case for 'Greeting'. Let's pick the topic "How's the weather?" and discuss possible questions for it.

Some questions might be:

1. "How's it feel out there today?" (if the Hero were inside)
2. "Isn't this great weather?" (if the weather was good, and the Hero was coming inside)
3. "Nice day, isn't it ma'am?" (if the weather was good, and the Hero was talking to his boss, who was a woman)

We could always have simply one statement or question for a topic, and make it generic enough to fit in all cases. However, your interactive book will be more entertaining and easy to read if you mix up the wording of the questions, and do so with intelligence.

Topics allow you to abstract one level from the statements or questions, so that the conversations in your game will sound more intelligent.

NOTE: Topics are used by Places (when a place goes to a conversation). In that place, you will define the people encountered, and the topics of conversation that can be discussed. You can have as many as 5 topics for the conversation of any place.

B. Statements

Statements/Questions are the phrases that the Hero can say for any given topic. A Statement/Question has the following components:

1. Statement Text
2. Topic linked to
3. Preconditions necessary for this statement to appear
4. Chance that it will appear, given that precondition is true

The Statement Text is parsed text (like Scene Text), which means that you can include information about the place, Hero, Encountered Person, etc. in the text. This text is what will appear to the Hero as an optional question for her to ask the Encountered Person.

The 'topic linked to' associates this statement/question with one of the topics. For example, "Wazzup?!?" would be the statement text, and it would be linked to the topic "Greeting".

Preconditions and Chance are used the same here as they have been used with the other data. All preconditions must be true for the statement text to appear. And if lots of statements are possible (all have preconditions that are true), then the Chance percentage will be used.

Let's use an example. We have a place called "Hallway" which has 3 possible people found (colleague, your boss, customer). This place links to a conversation with 2 topics ("Greeting" and "Help me"). We will start with the topic "Greeting".

There are three statements possible for "Greeting":

A. 1. Statement = "Good afternoon"
   2. Topic linked to = Greeting
   3. Precondition = Encountered Person's 'Title' = 'Your Boss'
   4. Chance = 100%

B. 1. Statement = "Wazzup?!?"
   2. Topic linked to = Greeting
   3. Precondition = Encountered Person's 'Title' = 'Colleague'
   4. Chance = 100%

C. 1. Statement = "How can I help you?"
   2. Topic linked to = Greeting
   3. Precondition = Encountered Person's 'Title' = 'Customer'
   4. Chance = 100%

So, when the Hero goes to the place "Hallway", one of three people will appear. Depending on who is encountered, the statement that the Hero can say will generated.

If the Hero encounters his boss, "Good afternoon" will be the option. If he encounters a customer, "How can I help you?" will be the option.

Same example. Now let's discuss the second topic, "Help me".

There are two statements possible for "Help me":

A. 1. Statement = "Excuse me. Could I ask you for a quick favor?"
   2. Topic linked to = Help me
   3. Precondition = Encountered Person's 'Title' does not equal 'Customer'
   4. Chance = 100%

B. 1. Statement = "Hello and welcome."
   2. Topic linked to = Help me
   3. Precondition = Encountered Person's 'Title' = 'Customer'
   4. Chance = 100%

So, when the Hero goes to Hallway, he will encounter someone, and have two options of what to say to that person. If he encounters his boss, his options will be:

1. Good Afternoon
2. Excuse me. Could I ask you for a quick favor?

If he encounters a customer, his options will be:

1. How can I help you?
2. Hello and welcome.

Notice that even though the topic is "Help me", the Hero will not ask the customer to help him, since it would probably not be an appropriate statement.

NOTE: Often you will have many statements in the same topic that have their preconditions met. In this case, Fun Studio will pick one from the group (using % chance). One and only one statement will appear for any given topic.

IMPORTANT: The Statement Editor has two parts. On top is the current statement that you are editing. Under it is a list of all other statements linked to the same topic. This is to help you see all of the possibilities for a particular topic.

FINAL NOTE: Always be sure that there is a statement and response for every situation. An easy way to do this is by putting in a default response and statement for each topic. The default would have no preconditions set up, and might have a chance of 1% (or something very small). By putting a small percentage chance, you will prevent it from appearing often when other statements or responses are possible.

If no response or statement is available, Fun Studio will put "You don't know what to say/do" for the statement, and nothing for the Response.

C. Responses

Responses are how the Encountered Person reacts to what the Hero says (in his statement or question). A response can be either something that the Encountered Person says or does. The data needed for a response is almost identical to that needed for statement.

A response has:

1. Response Text (also parsed, like Scene Text)
2. Topic linked to
3. Preconditions
4. Chance
5. Changes
6. Scene to call (one or none)

Just as with Statements, a Response has text that will appear based on the topic, preconditions, and the percentage chance. If a Response appears and it has changes set up, those changes will occur (just as a Scene causes changes). Finally, after the response is given, a scene may be called to continue the conversation. You will define that scene with the Responses Editor.

Let's continue our example. The place is Hallway, and the topics are "Greeting" and "Help me". The Hero can encounter his boss, a colleague, or a customer. In the previous chapter, we defined the statements possible. Below, we will define some possible responses.

A. 1. Text = "He nods and walks on."
   2. Topic linked to = Greeting
   3. Preconditions = Encountered Person's 'Title' = 'Your Boss'
   4. Chance = 100%
   5. Changes = None
   6. Scene to call = None

B. 1. Text = "Good to see you, buddy."
   2. Topic linked to = Greeting
   3. Preconditions = Encountered Person's 'Title' = 'Colleague'
   4. Chance = 100%
   5. Changes = None
   6. Scene to call = None

C. 1. Text = "No thank you."
   2. Topic linked to = Greeting
   3. Preconditions = Encountered Person's 'Title' = 'Customer'
   4. Chance = 50%
   5. Changes = None
   6. Scene to call = None

D. 1. Text = "Yes please. I was looking for a ..."
   2. Topic linked to = Greeting
   3. Preconditions = Encountered Person's 'Title' = 'Customer'
   4. Chance = 50%
   5. Changes = None
   6. Scene to call = Help customer find ...

E. 1. Text = "I'm too busy right now."
2. Topic linked to = Help me
3. Preconditions = Encountered Person's 'Title' does not equal 'Customer'
4. Chance = 30%
5. Changes = None
6. Scene to call = None

F. 1. Text = "Sure, I can help you."
2. Topic linked to = Help me
3. Preconditions = Encountered Person's 'Title' does not equal 'Customer'
4. Chance = 70%
5. Changes = None
6. Scene to call = Encountered Person helps Hero

OK. So now let's assume that the Hero goes to the place, encounters his boss, and says, "Good Afternoon". What would be the response?

In this case, the topic is Greeting, so the response must be A, B, C, or D. But only A meets the precondition (Encountered Person's 'Title' = 'Your Boss'). So the response would be, "He nods and walks on."

Second case. Hero encounters a Customer and says the Greeting "How can I help you?" What would the response be?

Again, the topic is Greeting, so it must be A, B, C, or D. Only C and D have the preconditions met, so Fun Studio would pick randomly between the two of them (50% chance for each). And if D were selected, a scene would be called after showing the response. NOTE: You must always put response text, even if the response calls a scene. This is because the response text will be shown first, and the user will hit the "Continue" button. Then the scene will be called.

Third case, Hero encounters a Customer and says the "Help me" topic, "Hello and welcome!" What would be the response...

In this case, the topic is "Help me", so E and F are possibilities. However, neither of them would be called, since their preconditions are not true. Therefore, there is no response possible for this situation, and the text would be blank.

NOTE: Always be sure that there is a statement and response for every situation. An easy way to do this is by putting in a default response and statement for each topic. The default would have no preconditions set up, and might have a chance of 1% (or something very small). By putting a small percentage chance, you will prevent it from appearing often when other statements or responses are possible.

If no response or statement is available, Fun Studio will put "You don't know what to say/do" for the statement, and nothing for the Response.

Chapter 22: Visiting characters in the game

Until now, if you wanted to talk to a person, you had to go to a place and hope to encounter them. But what if you want to visit one person in particular??

The Visit Editor allows the Author to define what characters can have direct contact with the Hero. We'll give an example...

Let's say that we have a person characteristic called "Hero has phone number". Each character in the game begins with "Hero has phone number" = False. But as the Hero goes through the game, she meets characters and discovers their phone numbers. Once she knows someone's phone number, a new list appears in the main screen, just beneath the list of places. And beside the list, there is text that says "Phone Call". If the hero clicks on a person, she can now see the picture of the person and call her.

This is all achieved through the Visit Editor. With the Visit Editor, you will define the following:

1. Can the Hero visit people?
2. If so, what is the Find Function used to determine who can be visited?
3. What are the scenes that can be called? (Links are used)
4. What does the text say that the Hero will see? ("Visit", "Phone Call", etc.)
5. What is the percentage chance that the person will be found?
6. Does the Visit call a Conversation?
7. If so, what preconditions must be true for the conversation to occur?
8. What topics can be discussed?

NOTE: A conversation will only occur if no scene is possible (either because none are set up, or none have their preconditions met).

To achieve the original example, we would have the Visit Editor defined as follows:

1. Can Hero visit = Yes
2. Find Function = Phone Number People
   (pulls from All People; precondition: The person's 'Hero has phone number' = True)
3. One scene in links = "Phone Call Scene"
4. Text = Phone Call
5. Chance = 100%
6. Conversation = No
7. Not applicable
8. Not applicable

The next sections will demonstrate the use of Visit. Notice that as you talk to people, and find out their phone numbers, they will appear in the list.

Chapter 23: Contemplation

In many games/interactive books, there will be information that you want the Hero to be able to know. For example, in this tutorial game, you can contemplate on yourself and see different person and game characteristics (those that are introspectable).

You might see the following:

Name = Tom
Dollars = 5
Apples = 2
Level Achieved = 3

What if you want to contemplate on other characters? You might want to remember who you have met, who was a bad person, or who was good...

The Contemplate Editor lets you set up all of these things. You must put the following information into the editor.

1. Hero can contemplate what? (on himself, on others, neither, or both)
2. If on himself, how many items can he contemplate at one time?
3. If on others, you need to set up the Contemplation Functions.
   These functions have four parts:
   A. Name of Contemplation Function
   B. Find Function used to make a list of people
   C. Text to put above
   D. Text to put below

If the Hero contemplates on himself, he will not see all the information in the person and game characteristics. First, he will only be able to see those characteristics which are introspectable. Second, he will only see as many as the Author defines (2, 3, 4, 5, or all). For example, if he can only see three, then he time he contemplates on himself, he will see a random grouping of three of the introspectable characteristics.

If the Hero can contemplate on others, you must define the Contemplation Functions. First, you will need to name the function. This name is what will be seen by the Hero when he chooses what to contemplate on.

Second, you will define a Find Function. For example, if the Hero contemplated on "People You Know", then the Find Function would be defined as the following:

1. Find Function = "People Hero has met"
2. Pull from: All People
3. Precondition: The Person's 'Has met Hero' = True

NOTE: Obviously, 'Has met Hero' would need to be a Person Characteristic. Further, each scene or response where the Hero meets another character would need to have a change (The Encountered Person's 'Has met Hero' = True).

Third, you will define the Text that goes above and below the list of people. Either or both of these could be blank.

Let's run through an example of a Contemplation Function.

1. Contemplation Function Name = People You Know
2. Find Function = People Hero has met
3. Text Above = "You have met the following people:"
4. Text Below = None

If the Hero were playing the game, he would select "People You Know" from the contemplation list, and click on "Go". He might then read:

"You have met the following people:

Julie, Ted, Al"

NOTE: When you exit this scene, you will be able to try this... This tutorial game keeps up with all the people that you have met, and allows you to contemplate on them.

Chapter 24: Events

Everything that we have talked about to this point has involved the Hero taking action. The Hero goes to a place, is in a scene or conversation, takes a test, visits someone, contemplates, etc... But what are the other characters doing while the Hero does all of this? What else is happening in this world?

Chapters 9 and 10 discuss Events and Interactions, the things that are happening in the world in parallel to the Hero's adventures. Events can occur to any person in the game (including the Hero). To create an Event (in the Events Editor), you need to enter the following information:

1. The name of the Event
2. When it occurs (after 5 days, after 2 weeks, etc.)
3. If it occurs every time (or only the first time)
4. Who might the event happen to (one fixed person or many people)?
5. If many people, what Find Function is used?
6. What changes will occur to the people?
7. Are all people in the group changed, or a random subset?
8. If random subset, what is the % chance that the event will happen?
9. How many people (maximum) could be affected?
10. How many people (minimum) could be affected?
11. Is this event news-worthy? (more on News in Chapter 11)

There is a lot of information here, so let's start with the easy stuff. #1, #2, and #3 are just like the Warnings Editor (Chapter 3). Basically, you name the Event and decide how often it will happen.

#4 and #5 define who the event might affect. Is it one fixed person or a group of people? If it's a group of people, then you'll use a Find Function to decide which characters are in the group. Also, if it's a group #7, 8, 9, and 10 will define how many people from that group are affected. The event might happen to all of them, or just a random portion. You, as the Author, will decide which.

#6 is the change that will occur to the person (or people). Changes here are just like the changes defined in Scenes or Responses.

#11 is where you define if the event is news-worthy. We'll discuss this more in Chapter 11. Basically, when an event is news-worthy, an article describing the event will be written into the Newspaper. The Hero can read the Newspaper to see what has happened.

Let's give an example of an event.

1. Event Name: "Lottery Small Winner"
2. Occurs after 2 days
3. Every time (so, after day 2, day 4, day 6, etc.)
4. Can happen to: All People
5. Find Function: All People
6. Changes: The Person's Money = Old Value + 1,000
7. Random subset
8. 1% chance (out of 100%)
9. 1 person maximum
10. 0 people minimum
11. News-worthy = Yes

Let's explore this example. Every 2 days, a lottery will occur in which all of the characters have a chance to win. If they win, they get $1,000 prize money. But there is only a 1% chance that anyone will win (pretty good odds, actually). There is also a maximum of one winner, and a minimum of 0 (meaning that no one would win). Finally, if someone wins, then an article might be written about it in the Newspaper.

Some notes. First, the % chance here is actually out of 100%. (This is different from Links, Items, Statements, and Responses, where the percentage chance is relative to the other possibilities.) In the Events Editor, 1% means 1%.

Second, the % chance can't be a fractional number, like 0.001% (which might have been better for the lottery). It has to be an integer number, like 1, 2, 3, etc. The next example shows how you could do the lottery with a bigger prize, and a smaller percentage chance.

Example 2: In this case, we'll use two events. The first is the same as above, with a modification to #6. The second event builds on the first.

1. Event Name: "Lottery Small Winner"
2. Occurs after 2 days
3. Every time (so, after day 2, day 4, day 6, etc.)
4. Can happen to: All People
5. Find Function: All People
6. Changes: The Person's Money = Old Value + 1,000
   The Person's 'Won Small Lottery' = True
7. Random subset
8. 1% chance (out of 100%)
9. 1 person maximum
10. 0 people minimum
11. News-worthy = Yes

1. Event Name: "Lottery Jackpot"
2. Occurs after 4 days
3. Every time
4. Can happen to: Small Lottery Winners
5. Find Function: (Precondition: The Person's 'Won Small Lottery' = True)
6. Changes: The Person's Money = Old Value + 1,000,000
7. Random subset
8. 1% chance (out of 100%)
9. 1 person maximum
10. 0 people minimum
11. News-worthy = Yes

Now we have two events, and the second one builds on the first. Only people who won the small lottery are entered into the Jackpot drawing. So now, the chance for a person to win the Jackpot is 1 in 10,000. (Still great odds on $1,000,000). This is all achieved by making an additional change in the first event, and using a Find Function in the second event.

Chapter 25: Interactions

Interactions occur between two characters in the game, causing changes to each of them. They can occur at any time (like Events), but they never include the Hero. To set up an Interaction, you must put in the following information:

1. Interaction Name
2. When it occurs (after 5 days, after 2 weeks, etc.)
3. If it occurs every time (or only the first time)
4. A Find Function to determine the first person
5. A Find Function to determine the second person
6. The % chance that the Interaction will occur

(And if the Interaction occurs...)

7. The changes that will happen to Person 1
8. The changes that will happen to Person 2
9. If the Interaction is news-worthy

This is very similar to Events. #1, 2, 3, 6, and 9 are exactly like components in the Events Editor. #4 and #5 determine who the possible people are (using Find Functions). #7 and #8 determine what changes will be made to each person (if the Interaction occurs).

NOTE: #6 (% chance) in Interactions will let you put in fractional numbers, like 0.04%, for small probability Interactions.

Let's discuss an example.

1. Interaction Name = Marriage
2. Occurs after 5 days
3. Every time
4. Find Function for Person 1: All Single Women
   (Pull from All People
   Precondition: The Person's Sex = Female
   The Person's Status does not equal Married)
5. Find Function for Person 2: All Single Men
   (Pull from All People
   Precondition: The Person's Sex = Male
   The Person's Status does not equal Married)
6. % chance = 0.1%
7. Change to Person 1 = The Person's Status = Married
   The Person's 'Married To' = Encountered Person
8. Change to Person 2 = The Person's Status = Married
   The Person's 'Married To' = Encountered Person
9. News-worthy = Yes

Here, we are using three important person characteristics (Sex, Status, and 'Married To') to help us define Find Functions and Changes. #4 will use the Find Function "All Single Women" to find those characters in the game that are both female and not married. #5 will use the Find Function "All Single Men" to find characters that are both male and not married.

Let's step through this process to understand what Fun Studio will do. Let's imagine that the function "All Single Women" returns Julie, Rachel, Monica, and Luisa. And, the function "All Single Men" returns Patrick, Tom, Ross, Chandler, and Ravi. Fun Studio will then put both lists into a random order. It might get "Rachel, Luisa, Julie, and Monica" for women, and "Ross, Patrick, Tom, Chandler, and Ravi" for men.

With the randomized lists, it will then look for interactions:

1. Rachel and Ross? 0.1% chance... No.
2. Luisa and Patrick? 0.1% chance... No.
3. Julie and Tom? 0.1% chance... No.
4. Monica and Chandler? O.1% chance... No.

Since there were no more women to use, this Interaction would be over, with no marriages occuring. Notice how Ravi was left off, since he randomly was put at the end, and there were not enough women. However, he'll have another chance in five days. The lists will be put in random order again, and Fun Studio will look for more interactions.

Let's imagine that a Marriage did occur, between Rachel and Ross... In this case, the changes would occur to both, so that both had 'Status' = Married. Because of this change, the next time the Interaction occurs, neither would be eligible for marriage.

NOTE: This example uses the same changes for both people, but often you would want to make different changes for each person.

Chapter 26: News

News can be called by either Events or Interactions. News is made up of three components:

1. Headline
2. Parsed Text
3. Precondition

The Headline is the main sentence of the Event or Interaction. It appears at the top of the article.

The Parsed Text appears for each person who experienced the event or interaction. (In the case of an Interaction, parsed text will include a Person 1 and Person 2).

The Precondition defines when news will occur. Even if an event happens, and is set as news-worthy, it won't automatically be put in the newspaper. The preconditions must be true for it to be put there.

Let's give an example of news for an Event. We'll use the Lottery Event set up in Chapter 9. We will also set up news for this event in the following way:

1. Headline = "Big $1,000 Winner!!"
2. Parsed Text = "--Event-Name-- won $1,000 in the lottery!!"
3. Precondition: The Person's 'Knows Hero' = True

So in this example, the only way that the news will be printed is if the person who won knows the Hero (set by the precondition). Also notice that the parsed text says --Event-Name--. In other parsed text examples, it might say --Pe-Encountered Person-Name-- or --Pe-Hero-Age--. But since an event can only happen to one person, Fun Studio just needs to see --Event-, and it knows that we are talking about the person affected by the event.

So, in this example, let's imagine that two winners were declared (we would need to change the Event to have a maximum of 2 people). Let's also imagine that those two people, Julie and Tom, knew the Hero. If this were the case, the newspaper might read:

"Big $1,000 Winner!!
Julie won $1,000 in the lottery!!
Tom won $1,000 in the lottery!!"

So you see, the headline appears once, and the parsed text appears for each person that won.

The only difference in News for Events and News for Interactions is the second person involved in an Interaction. The second person affects both Parsed Text and Preconditions. Let's give an example. Assume the Interaction of Marriage from the previous chapter. Now let's define News for Marriage in the following way:

1. Headline: "Wedding Bells!!"
2. Parsed Text: "--Int1-Name-- got married to --Int2-Name--!! What a celebration!"
3. Precondition: The Person's 'Knows Hero' = True

Parsed Text now uses --Int1- and --Int2- for Person 1 and Person 2. The precondition applies to both people, but only needs to be true for one of them. So let's look at four examples:

1. Julie and Tom got married. Both knew the Hero.
2. Luisa and Patrick got married. Neither knew the Hero.
3. Rachel and Ross got married. Rachel knew the Hero.
4. Monica and Chandler got married. Chandler knew the Hero.

In these four cases, #1, 3, and 4 would go in the newspaper, because at least one of the people knew the Hero (and met the Precondition).